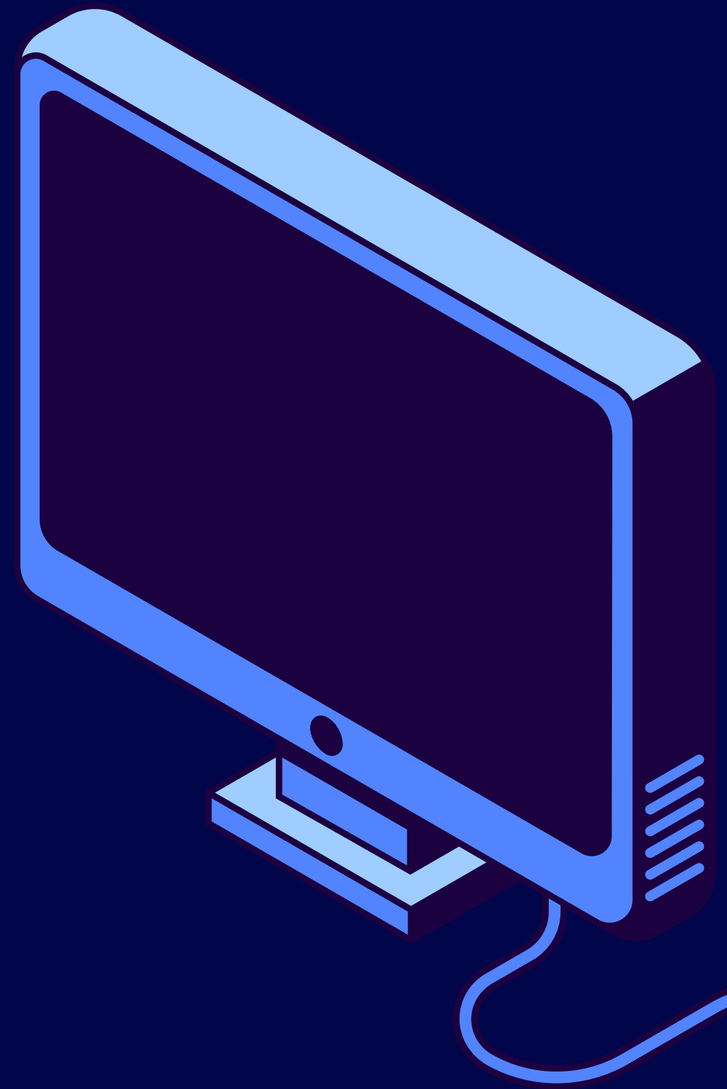
Sistemas Operativos

Ayudantía 2: Pipes y Tarea 1



Profesores: Víctor Reyes, Yerko Ortiz
Ayudante: Diego Banda

Contacto

Sección 1 y 3

diego.banda@mail.udp.cl

darKlouds

github.com/DiegoBan/SO2025-1

Corrector en Sección 3

alex.marambio@mail.udp.cl

Alex.marambio

Sección 1: Profesor Yerko Ortiz
Sección 3: Profesor Víctor Reyes

Variables entre procesos

```c
int main(){
    int contador = 0;
    pid_t pid = fork();

        if(pid > 0){
            for(int i = 0;i < 10 ; i++){
            contador++;
            printf("Soy el proceso padre y llevo la cuenta de %d\n",contador);
            }
        }else if(pid == 0){
            for(int i = 0;i < 10 ; i++){
            contador++;
            printf("Soy el proceso Hijo y llevo la cuenta de %d\n", contador);
            }
        }
    return 0;
}
```

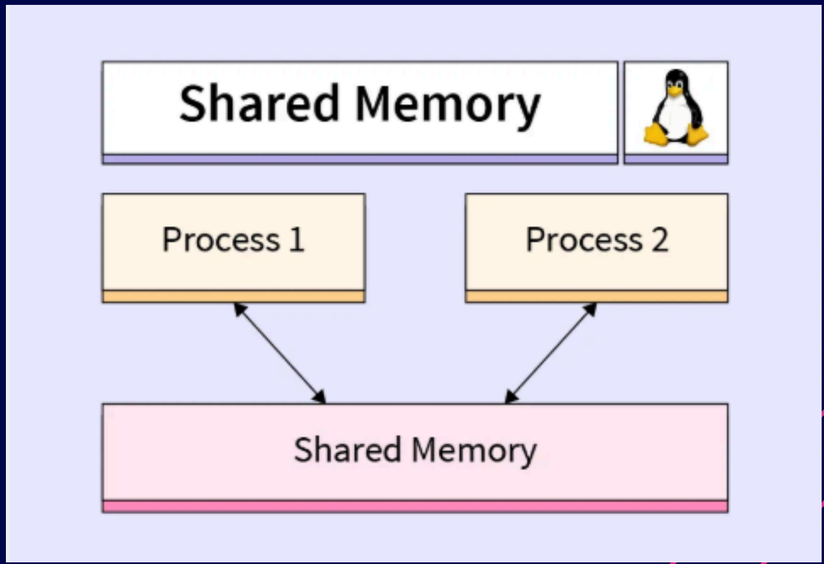
¿Los procesos padre e hijo comparten el contador? ¿O cada uno tiene uno propio?

```
Soy el proceso padre y llevo la cuenta de 1
Soy el proceso Hijo y llevo la cuenta de 1
Soy el proceso padre y llevo la cuenta de 2
Soy el proceso Hijo y llevo la cuenta de 2
Soy el proceso Hijo y llevo la cuenta de 3
Soy el proceso padre y llevo la cuenta de 3
Soy el proceso Hijo y llevo la cuenta de 4
Soy el proceso padre y llevo la cuenta de 4
Soy el proceso Hijo y llevo la cuenta de 5
Soy el proceso padre y llevo la cuenta de 5
```

Cada proceso tiene un contador propio, que no afecta al otro.

¿Cómo Podemos compartir variables?

Mediante **Memoria Compartida:**
Una forma de comunicacion entre los procesos , donde se permite compartir memoria entre ellos.



- shmget: se utiliza para crearo acceder a un espacio de memoria compartida, necesita una llave, un tamaño y opciones.
- shmid: es el id de la memoria.
- shmaddr: se utiliza para especificar la dirección de la memoria, donde se conecta el proceso.
- shmctl: se utiliza para realizar operaciones de control en la memoria.
- shmdt: desconecta la memoria de un proceso.

Ejemplo Memoria compartida

```c
key_t key = 1234;//Se establece llave del espacio de memoria
int shmid;//espacio del memoria

shmid = shmget(key,sizeof(int), IPC_CREAT | 0666);
if(shmid == -1) {//Manejo de errores
    perror("shmget");
    exit(EXIT_FAILURE);
}
int *contador = (int*) shmat(shmid, NULL, 0);//Creacion de la variable
if(contador == (int*) -1) { // manejo de errores
    perror("shmat");
    exit(1);
}
*contador = 0;//Inicio en 0
```

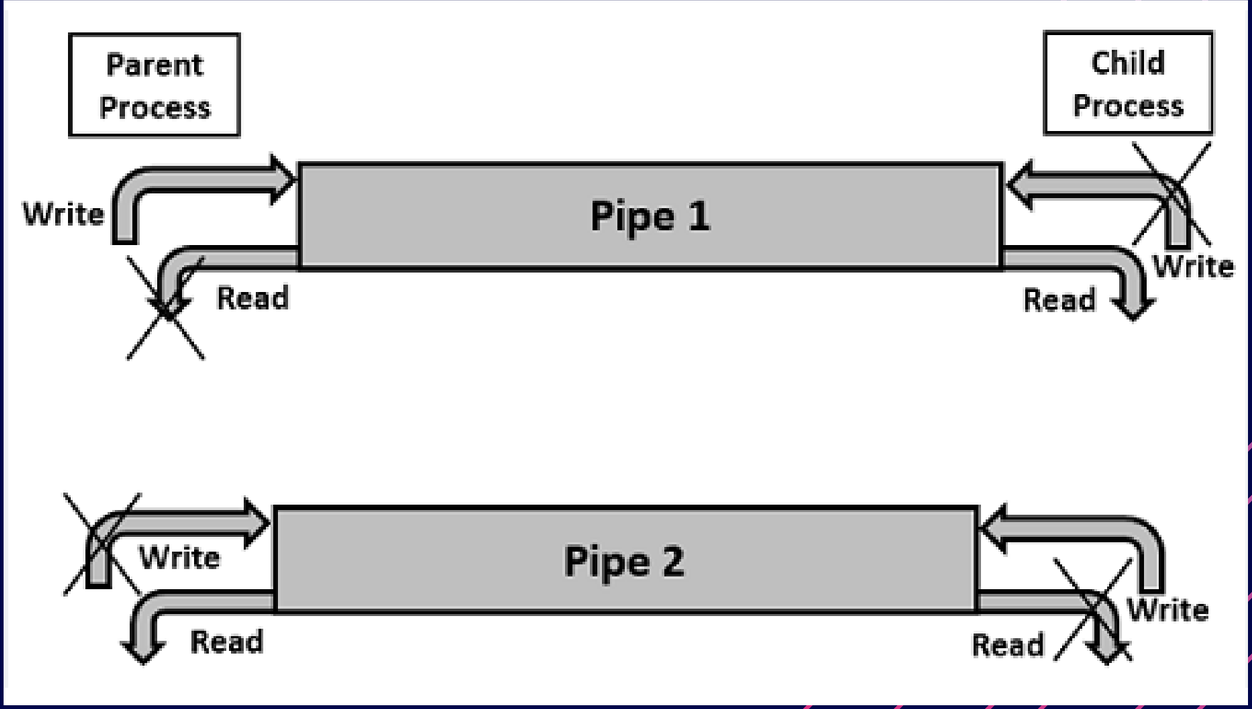
```c
        for(int i = 0;i < 10 ; i++){
        (*contador)++;
        printf("Soy el proceso padre y llevo la cuenta de %d\n",*contador);
        }
    }else if(pid == 0){
        for(int i = 0;i < 10 ; i++){
        (*contador)++;
        printf("Soy el proceso Hijo y llevo la cuenta de %d\n",*contador);
        }
        if(shmdt(contador) == -1){
            perror("shmdt");
            exit(EXIT_FAILURE);
        }
        if(shmctl(shmid, IPC_RMID,NULL) == -1){
            perror("shmctl");
            exit(EXIT_FAILURE);
```

Resultado

```
Soy el proceso padre y llevo la cuenta de 1
Soy el proceso Hijo y llevo la cuenta de 2
Soy el proceso padre y llevo la cuenta de 3
Soy el proceso Hijo y llevo la cuenta de 4
Soy el proceso padre y llevo la cuenta de 5
Soy el proceso Hijo y llevo la cuenta de 6
Soy el proceso padre y llevo la cuenta de 7
Soy el proceso Hijo y llevo la cuenta de 8
Soy el proceso padre y llevo la cuenta de 9
Soy el proceso Hijo y llevo la cuenta de 10
Soy el proceso padre y llevo la cuenta de 11
Soy el proceso Hijo y llevo la cuenta de 12
```

Pipes

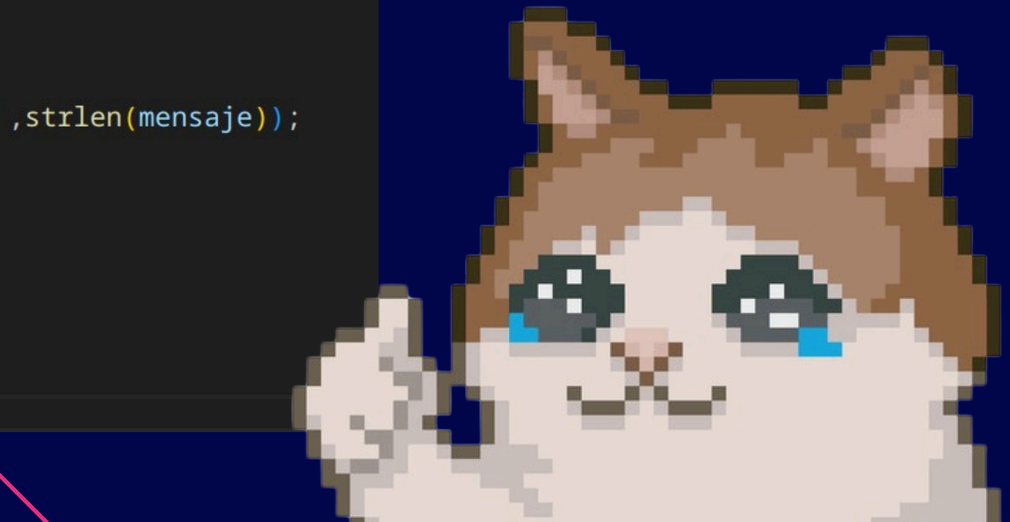
Pipes anónimos: Tienen una comunicación unidireccional entre dos procesos, se utiliza entre los procesos padre e hijo.



- Para crear pipes se hace atreves de la syscall pipe(), se pasa un array de dos elementos tipo int.
- se utiliza 0 para leer (READ) y 1 para escribir (WRITE).
- Las dos syscall necesitan :
  Archivo que represente al pipe.
  Espacio de memoria.
  Tamaño.

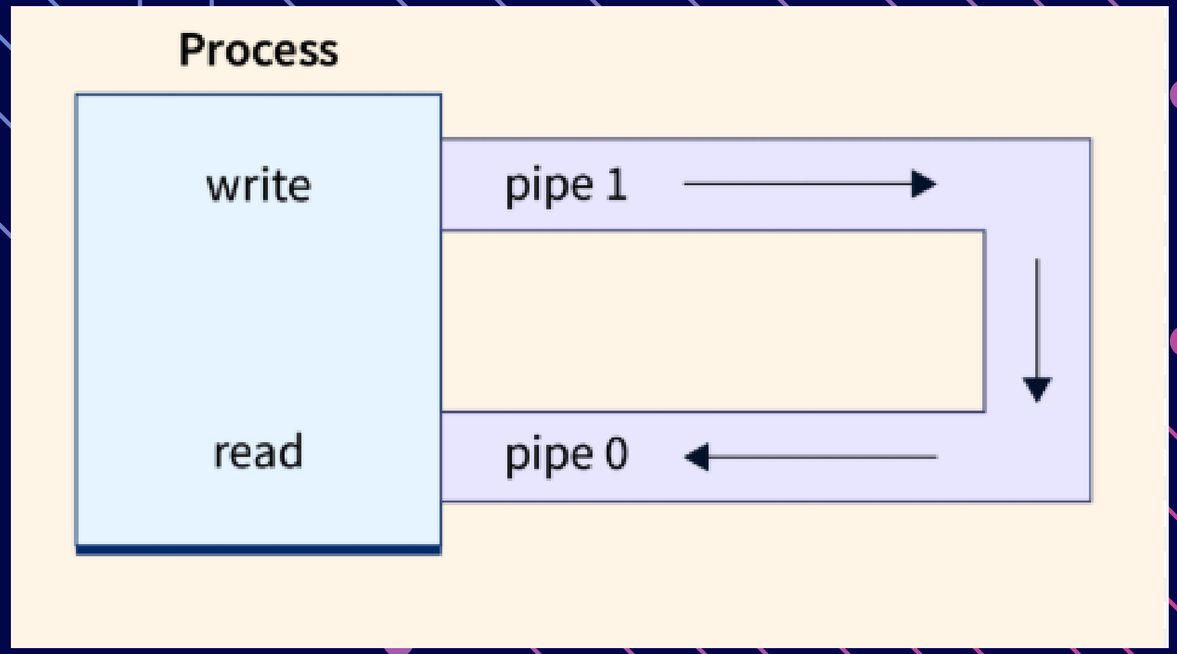
Ejemplo Pipes sin nombre

```c
int main(){
    int fd1[2];
    pid_t pid;
    pipe(fd1);// Creacion de pipes
    pid = fork();

    if(pid == 0){
        close(fd1[1]);
        char buffer[100];
        int Buffer_read = read(fd1[0], buffer, sizeof(buffer));
        if(Buffer_read == -1){// Manejo de errores
            perror("READ PIPE");
            exit(-1);
        }
        buffer[Buffer_read] = '\0'; // señal de termino
        printf("LLegaron las %s\n",buffer);
    } else {
        close(fd1[0]);
        char *mensaje = "Pipshass";
        ssize_t bytes_escrito = write(fd1[1],mensaje ,strlen(mensaje));
        if(bytes_escrito == -1){
            perror("WRITE PIPE");
            exit(-1);
        }
        printf("Enviado\n");
        close(fd1[1]);
    }
}
```



Pipes con nombre

Pipes FIFO (Con nombre) : Tienen una comunicación bidireccional, no requieren una ser padre e hijo.



- Se crean con la syscall mkfifo(), se le tiene que dar un nombre y respectivos permisos en octal.
- Tiene read() y write()
- Eliminar al final de la ejecucion el pipe, ya que este permanece luego de haber finalizado la ejecución.

Deben utilizarlo en la tarea

Ejemplo Pipes con Nombre

```c
int main(){
    const char *fifo_path = "./my_fifo2";// Nombre de la ruta del pipe

    int fd;
    int num;

    mkfifo(fifo_path,0666); // Creacion del pipe

    fd = open(fifo_path, O_WRONLY); // Abertura del pipe

    printf("Ingrese un numero para multiplicar por 100\n");
    scanf("%d", &num);

    write(fd, &num, sizeof(num)); // Envio de mensaje

    close(fd);
    unlink(fifo_path);

    return 0 ;
}
```

Codigo 1

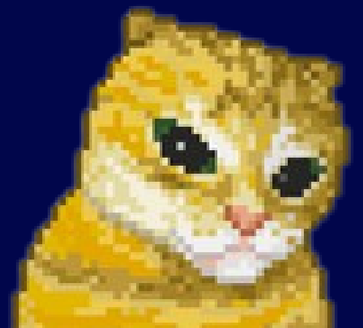
```c
int main(){
    const char *fifo_path = "./my_fifo2";// Nombre de la ruta

    int fd;
    int num;

    mkfifo(fifo_path,0666); // Creacion del pipe

    fd = open(fifo_path, O_WRONLY); // Abertura del pipe

    read(fd, &num, sizeof(num)); // recibo del mensaje

    printf("su numero multiplicado por 100 es: %d\n", num);

    close(fd);
    unlink(fifo_path);

    return 0 ;
```

Codigo 2

Revisar

Tarea 1

Rubrica



¡Gracias por su atención!